

LOGIC CIRCUITS

Focus:

The objective of this lab is to devise and piece together a series of binary logic elements to effect an ultimate mathematical operation such as addition, subtraction, or multiplication.

Overview:

Information can be represented and stored on a variety of electrical/mechanical devices. In many cases, the information relates to measurable variables such as elapsed time, total rainfall, or accumulated charge (for which the hourglass, rain gauge, and capacitor, respectively, are suitable representation devices.) But what about abstract information such as quantities in mathematics? Here we create an analogy between something that can be stored and measured in an electrical/mechanical device and a mathematical value. For example, we can assign an equivalence between mathematical value and electrical charge. The extent to which we can operate on that electrical charge via the capacitor and our measuring instruments is the degree to which we can perform analogous mathematical calculations.

There are problems, however, with relating mathematics to storable parameters on physical devices. Two of the more important ones are:

- 1) Measuring instruments are rarely more accurate than three decimal digits, so mathematics carried out through these devices would have intrinsic limited accuracy.
- 2) There is typically unrecoverable loss of information. For example, a capacitor could leak away part of its charge, i.e., its analogous mathematical value would arbitrarily change.

However, there is a solution to these problems: store mathematical values in discrete rather than analog form. Here one uses devices whose variations are limited to discrete states, typically two – on or off, positive or negative, closed or open. Then, by representing mathematical quantities in a number system having only two digits – a binary number system – any value can be represented with arbitrary accuracy by linking together a sequence of two-state devices and setting the appropriate state for each device. Information integrity in this discrete representation is better than that of analog representation because information loss requires an arbitrary change of state of a device, not a drift in value. That is much less likely, and there are ways to correct for it.

Computers are developed to store and mathematically manipulate quantitative information. Earlier computers were analog, usually electrical or mechanical. Circuits and mechanisms were built to represent fixed mathematical problems with results appearing in the form of a final voltage or a rotation angle of a gear. The electrical and mechanical equivalents of addition, multiplication, integration, and differentiation were

incorporated into these computers. Complex problems could be solved, albeit with the accuracy problems mentioned above.

Then, the digital computer emerged. In these systems, the electrical and mechanical analogies for mathematical operations are replaced by the digital manipulation of 1's and 0's – the two possible states in binary devices storing information. Logic circuits are used to carry out mathematics with binary devices.

Procedure:

This lab is a virtual lab and can be found under **Logical Circuits** at www.jhu.edu/~virtlab/virtlab.html.

It is necessary and sufficient to consider logic elements for which there are two binary inputs and one binary output. We consider three logic elements from which all binary logic may be constructed: the AND, OR and NOT gates.

In the problems that follow, use the Circuit Builder to develop your circuits. When you link to the Circuit Builder, you will be asked to specify the number of inputs and outputs. This will be determined by the specific problem on which you are working. You will then be presented with the circuit display having the specified number of inputs and outputs.

To build your circuit, drag logic components into the circuit area. Inputs (circles) may be connected to outputs (squares) by clicking first on one, then dragging to the other and releasing the mouse button. A single output (square) may be connected to several inputs (circles), but only one output may be connected to any input.

Once your circuit is completed, you can choose either the “single input” or “all inputs” option. With the “single input” option, you can specify a particular bit configuration of inputs. Once these inputs are specified, the state (1 or 0) of each logic line will be displayed. This will give you an opportunity to find errors in your circuit. With the “all possible inputs” option, outputs for all possible inputs will be presented. This should be your final confirmation that your circuit operates correctly.

The chart below shows the available logic elements, their graphic representation, and their input/output tables.

AND		OR		NOT		NAND		NOR	
Input	Output	Input	Output	Input	Output	Input	Output	Input	Output
0 0	0	0 0	0	0	1	0 0	1	0 0	1
0 1	0	0 1	1	1	0	0 1	1	0 1	0
1 0	0	1 0	1			1 0	1	1 0	0
1 1	1	1 1	1			1 1	0	1 1	0

Problems:

1. Confirm that each of the elementary circuits behaves as advertised. That is, use the Circuit Builder to establish the logic behavior of AND, NOT, OR, NAND, and NOR. These will all be two-input and one-output circuits (except for NOT, which will be one-input and one-output.)
2. Using only the component NAND, create a circuit that is equivalent to the logic of an AND gate. Do the same to emulate a NOT gate. Do the same to emulate an OR gate. (If this is possible, it means that all binary logic circuitry can ultimately be generated from the single logic element NAND.)
3. Using only the component NOR, see if you can create circuits equivalent to AND, OR, and NOT as you did in Problem 2.
4. Using only AND, NOT, and OR, produce a three-input AND circuit, i.e., the output is 0 unless all three inputs are a 1. (You do not have to use all three circuit elements.)
5. Using only AND, NOT, and OR, produce a three-input OR circuit, i.e., the output is 1 if any of the inputs is 1.
6. Create a two-input “adder” with two outputs: the one-digit result of the add and the value of a “carry” bit. In binary arithmetic, $1+1=0$ with a carry=1.
7. One of the more interesting public works problems is the “Superbowl” problem. At the beginning of halftime during the Superbowl, 35 million toilets are flushed almost simultaneously. The resulting loss of water pressure wreaks havoc on many municipal water systems. Here you will solve the problem for a “three toilet” system. Devise a logic circuit whose “1” inputs represent “flushes” and whose “1” outputs represent opened water-feed valves. If no more than one toilet is flushed, that toilet’s water valve opens, and the others remain closed. If more than one toilet is flushed, all the water valves remain closed.
8. You are designing a robot to move toward a light source. Three photosensors S_L , S_C , and S_R are mounted at the front of the robot pointing 45° to the left, straight ahead, and 45° to the right, respectively. Two wheels W_L and W_R are powered depending on the output of the sensors. If S_L detects light, the robot is pointing too far to the right, and the right wheel W_R must be powered up to turn the robot to the left. The opposite is necessary if S_R receives light. If only the forward-pointing sensor S_C is lit, then both wheels W_L and W_R should be powered to propel the robot forward. If none of the sensors sees light, the robot should turn.

If one treats the sensors as having binary outputs, i.e., either “on” or “off”, and the powered wheels as being “on” or “off”, a simple logic circuit can be used to actuate

the wheels under each sensor condition. Create such a logic circuit using only NAND gates, and using the least number of these.

9. This picture is a schematic diagram of a 14-pin CMOS chip that contains 4 NAND gates. Your task is to design a printed circuit board that implements the robot circuit you produced in Problem 8. Using as many CMOS chips as necessary, and taking into consideration which pins constitute inputs and outputs, create the circuit by connecting the appropriate pins. But now, try to route the connections so that the “wires” do not cross one another. This is a typical problem in printed circuit design. The objective is to etch the circuit onto a single layer of a copper-coated circuit board which contains pin holes to accommodate the chips. (Note: This problem does not require the use of the Circuit Builder. Just sketch out the connections between the chip(s).)

